МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Универсальные алгебры и алгебра отношений

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студента 3 курса 331 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Яхина Шамиля Илдусовича

Преподаватель
профессор, д.ф.-м.н. ___________ В. А. Молчанов

подпись, дата

Саратов 2022

СОДЕРЖАНИЕ

1 Теория 4
1.1 Алгебраические операции 4
1.2 Основные операции над бинарными отношениями 4
1.3 Основные операции над матрицами 4
2 Результаты работы 6
2.1 Алгоритм проверки операции на ассоциативность 6
2.2 Алгоритм проверки операции на коммутативность 6
2.3 Алгоритм проверки операции на идемпотентность 6
2.4 Алгоритм проверки операции на обратимость 7
2.5 Алгоритм проверки операции на дистрибутивность 7
2.6 Алгоритм операции объединения для бинарных отношений 8
2.7 Алгоритм операции пересечения для бинарных отношений 8
2.8 Алгоритм операции дополнения для бинарного отношения 8
2.9 Алгоритм операции произведения для бинарных отношений 9
2.10 Алгоритм операции нахождения обратного для бинарного отношения 9
2.11 Алгоритм операции сложения матриц над конечным полем 10
2.12 Алгоритм операции умножения матриц над конечным полем 10
2.13 Алгоритм операции транспонирования матрицы над конечным полем 11
2.14 Алгоритм нахождения определителя матрицы над конечным полем 11
2.15 Алгоритм операции обращения матрицы над конечным полем 12
3 Код программы 14
4 Результаты тестирования программ 33
ЗАКЛЮЧЕНИЕ 41

Цель работы — изучение основных понятий универсальной алгебры и операций над бинарными отношениями.

1 Теория

1.1 Алгебраические операции

Отображение $f : A^n \to A$ называется алгебраической n-арной операцией или просто алгебраической операцией на множестве A. При этом n называется порядком или арностью алгебраической операции f.

Бинарная операция $\cdot$ на множестве A называется:

1. ассоциативной, если $\forall x, y, z \in A$ выполняется равенство:
   $x \cdot (y \cdot z) = (x \cdot y) \cdot z$;
2. коммутативной, если $\forall x, y \in A$ выполняется равенство:
   $x \cdot y = y \cdot x$;
3. идемпотентной, если $\forall x \in A$ выполняется равенство:
   $x \cdot x = x$;
4. обратимой, если $\forall x, y \in A$ уравнения $x \cdot a = y$ и $b \cdot x = y$ имеют решение, причем единственное;
5. дистрибутивной относительно операции $+$, если $\forall x, y, z \in A$ выполняются равенства
   $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$,
   $(y + z) \cdot x = (y \cdot x) + (z \cdot x)$;

1.2 Основные операции над бинарными отношениями

1. Теоретико-множественные операции $(\cup, \cap, \neg)$
2. *Обращение* бинарных отношений: обратным для бинарного отношения $\rho \subset A \times B$ называется бинарное отношение $\rho^{-1} \subset B \times A$, определяющееся по формуле:
   $\rho^{-1} = \{(b, a) : (a, b) \in \rho\}$.
3. *Композиция* бинарных отношений: композицией бинарных отношений $\rho \subset A \times B$ и $\sigma \subset B \times C$ называется бинарное отношение $\rho\sigma \subset A \times C$, определяющееся по формуле:
   $\rho\sigma = \{(a, c) : (a, b) \in \rho$ и $(b, c) \in \sigma$ для некоторого $b \in B\}$.

1.3 Основные операции над матрицами

Основными операциями для матриц являются следующие операции:

Все операции выполняются над конечным полем.

1. Сложения;

Суммой матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинаковой размерности $m \times n$ называется третья матрица $C = (c_{ij})$ такой же размерности $m \times n$, где ее элементы c_{ij} определяются равенством $c_{ij} = (a_{ij} + b_{ij})$ для всех значений индексов.

2. Умножения;

   Произведением матриц $A = (a_{ij})$ размерности $m \times n$ и $B = (b_{ij})$ размерности $n \times p$ называется третья матрица $C = (c_{ij})$ размерности $m \times p$, где элемент c_{ij} = a_{i1} * b_{1j} + ... + a_{in} * b_{nj} для всех значений индексов $i = 1, 2, ..., m$, $j = 1, 2, ..., p$.

3. Транспонирования;

   Матрица B размера $n \times m$ называется транспонированной по отношению к матрице A размера $m \times n$, если k– й столбец матрицы B состоит из элементов k– й строки матрицы A, для всех $k = 1, 2, ..., m$.

4. Обращения;

   Матрица B называется обратной по отношению к матрице A, если $A * B = B * A = E$.

Более подробнее эти операции будут рассмотрены при расписывании алгоритмов в следующем пункте.

2 Результаты работы

2.1 Алгоритм проверки операции на ассоциативность

Вход: Размерность матрицы N, множество элементов mn из таблицы и таблица Кэли keli, представленная матрицей размерности $N \times N$.

Выход: "Операция ассоциативна"или "Операция не ассоциативна".

1. bool prov = true;
2. Запускается цикл for с x от 0 до N - 1.
   а) Запускается цикл for с y от 0 до N - 1.
      i. Запускается цикл for с z от 0 до N - 1 и если в нем хоть раз выполняется keli[x][keli[y][z]] != keli[keli[x][y]][z], prov присваивается false.
3. Если prov = true, выводится "Операция ассоциативна иначе "Операция не ассоциативна"

Временная сложность алгоритма проверки операции на ассоциативность = $O(n^3)$

2.2 Алгоритм проверки операции на коммутативность

Вход: Размерность матрицы N, множество элементов mn из таблицы и таблица Кэли keli, представленная матрицей размерности $N \times N$.

Выход: "Операция коммутативна"или "Операция не коммутативна".

1. bool prov = true;
2. Запускается цикл for с x от 0 до N - 1.
   а) Запускается цикл for с y от 0 до N - 1 и если в нем хоть раз выполняется keli[x][y] != keli[y][x], prov присваивается false.
3. Если prov = true, выводится "Операция коммутативна иначе "Операция не коммутативна"

Временная сложность алгоритма проверки операции на коммутативность = $O(n^2)$

2.3 Алгоритм проверки операции на идемпотентность

Вход: Размерность матрицы N, множество элементов mn из таблицы и таблица Кэли keli, представленная матрицей размерности $N \times N$.

Выход: "Операция идемпотентна"или "Операция не идемпотентна".

1. bool prov = true;

2. Запускается цикл for с x от 0 до N - 1 и если в нем хоть раз выполняется keli[x][x] != mn[x], prov присваивается false.
3. Если prov = true, выводится "Операция идемпотентна иначе "Операция не идемпотентна"

Временная сложность алгоритма проверки операции на идемпотентность = $O(n)$

2.4 Алгоритм проверки операции на обратимость

Вход: Размерность матрицы N, множество элементов mn из таблицы и таблица Кэли keli, представленная матрицей размерности $N \times N$.

Выход: "Операция обратима"или "Операция не обратима".

1. bool prov = true;
2. Запускается цикл for с x от 0 до N - 1.
   а) Запускается цикл for с y от 0 до N - 1 и если в нем хоть раз выполняется keli[x][y] != 1 и keli[y][x] != 1, prov присваивается false.
3. Если prov = true, выводится "Операция обратима иначе "Операция не обратима"

Временная сложность алгоритма проверки операции на обратимость = $O(n^2)$

2.5 Алгоритм проверки операции на дистрибутивность

Вход: Размерность матрицы N, множество элементов mn из таблицы и две таблицы Кэли keli1 и keli2, представленные матрицами размерности $N \times N$.

Выход: "Операция дистрибутивна"или "Операция не дистрибутивна".

1. bool prov = true;
2. Запускается цикл for с x от 0 до N - 1.
   а) Запускается цикл for с y от 0 до N - 1.
      i. Запускается цикл for с z от 0 до N - 1 и если в нем хоть раз выполняется keli1[x][keli2[y][z]] != keli2[keli1[x][y]][keli1[x][z]] или keli1[keli2[y][z]][x] != keli2[keli1[y][x]][keli1[z][x]], prov присваивается false.
3. Если prov = true, выводится "Операция дистрибутивна иначе "Операция не дистрибутивна"

Временная сложность алгоритма проверки операции на дистрибутивность = $O(n^3)$

2.6 Алгоритм операции объединения для бинарных отношений

Вход: Размерность матриц $N \times M$, два бинарных отношения matrix1 и matrix2, представленные матрицами размерности $N \times M$.

Выход: Матрица с выполненной операцией объединения.

1. Создается булева матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N - 1.
   а) Запускается цикл for с j от 0 до M - 1 и если в нем выполняется matrix1[i][j] = 1 или matrix2[i][j] = 1, matrix_res[i][j] присваивается 1, иначе matrix_res[i][j] присваивается 0.
3. Выводится матрица matrix_res.

Временная сложность алгоритма операции объединения для бинарных отношений = $O(n^2)$

2.7 Алгоритм операции пересечения для бинарных отношений

Вход: Размерность матриц $N \times M$, два бинарных отношения matrix1 и matrix2, представленные матрицами размерности $N \times M$.

Выход: Матрица с выполненной операцией пересечения.

1. Создается булева матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N - 1.
   а) Запускается цикл for с j от 0 до M - 1 и если в нем выполняется matrix1[i][j] = 0 или matrix2[i][j] = 0, matrix_res[i][j] присваивается 0, иначе matrix_res[i][j] присваивается 1.
3. Выводится матрица matrix_res.

Временная сложность алгоритма операции пересечения для бинарных отношений = $O(n^2)$

2.8 Алгоритм операции дополнения для бинарного отношения

Вход: Размерность матрицы $N \times M$, бинарное отношение matrix, представленное матрицей размерности $N \times M$.

Выход: Матрица с выполненной операцией дополнения.

1. Создается булева матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N - 1.

   а) Запускается цикл for с j от 0 до M - 1 и если в нем matrix[i][j] = 1, matrix_res[i][j] присваивается 0, иначе matrix_res[i][j] присваивается 1.
3. Выводится матрица matrix_res.

Временная сложность алгоритма операции дополнения для бинарного отношения = $O(n^2)$

2.9 Алгоритм операции произведения для бинарных отношений

*Вход:*Размерности первой матрицы $N \times M$, размерности второй матрицы $M \times P$, два бинарных отношения matrix1 и matrix2, представленные матрицами размерности $N \times M$ и $M \times P$.

Выход: Матрица с выполненной операцией произведения.

1. Создается булева матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N - 1.
   а) Запускается цикл for с j от 0 до P - 1
      i. Запускается цикл for с k от 0 до M - 1 и если в нем выполняется matrix1[i][k] = 1 и matrix2[k][j] = 1, matrix_res[i][j] увеличивается на 1 (но т.к. это бинарная матрица, дальше единицы увеличиваться не будет).
3. Выводится матрица matrix_res.

Временная сложность алгоритма операции произведения для бинарных отношений = $O(n^3)$

2.10 Алгоритм операции нахождения обратного для бинарного отношения

Вход: Размерность матрицы $N \times M$, бинарное отношение matrix, представленное матрицей размерности $N \times M$.

Выход: Матрица с выполненной операцией нахождения обратного.

1. Создается булева матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N - 1.
   а) Запускается цикл for с j от 0 до M - 1 и в нем matrix_res[i][j] присваивается matrix[j][i].
3. Выводится матрица matrix_res.

Временная сложность алгоритма операции дополнения для бинарного отношения = $O(n^2)$

2.11 Алгоритм операции сложения матриц над конечным полем

Вход: Размерности первой матрицы $N1 \times M1$, размерности второй матрицы $N1 \times M1$, две матрицы matrix1 и matrix2 размерности $N1 \times M1$ и $N1 \times M1$ и число, по модулю которого будет выполняться операция mat_mod.

Выход: Матрица с выполненной операцией сложения над конечным полем.

1. Создается матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N1 - 1.
   а) Запускается цикл for с j от 0 до M1 - 1 и в нем matrix_res[i][j] присваивается matrix1[i][j] + matrix1[i][j].
3. Каждый элемент matrix_res берется по модулю mat_mod.
4. Выводится матрица matrix_res.

Временная сложность алгоритма операции сложения матриц над конечным полем = $O(n * m)$, где n это N1, m это M1

2.12 Алгоритм операции умножения матриц над конечным полем

Вход: Размерности первой матрицы $N1 \times M1$, размерности второй матрицы $N2 \times M2$, две матрицы matrix1 и matrix2 размерности $N1 \times M1$ и $N2 \times M2$ и число, по модулю которого будет выполняться операция mat_mod.

Выход: Матрица с выполненной операцией умножения над конечным полем.

1. Если M1 != N2, алгоритм не выполняется.
2. Создается матрица matrix_res, которая заполняется следующим образом:
3. Запускается цикл for с i от 0 до N1 - 1.
   а) Запускается цикл for с j от 0 до M2 - 1
      i. Запускается цикл for с k от 0 до M1 - 1 и в нем каждый раз matrix_res[i][j] увеличивается на matrix1[i][k] * matrix2[k][j].
4. Каждый элемент matrix_res берется по модулю mat_mod.
5. Выводится матрица matrix_res.

Временная сложность алгоритма операции умножения матриц над конечным полем = $O(n * m2 * m1)$, где n - это N1, m1 - это M1, m2 - это M2.

2.13 Алгоритм операции транспонирования матрицы над конечным полем

Вход: Размерность матрицы $N1 \times M1$, матрица matrix размерности $N1 \times M1$ и число, по модулю которого будет выполняться операция mat_mod.

Выход: Матрица с выполненной операцией транспонирования над конечным полем.

1. Создается матрица matrix_res, которая заполняется следующим образом:
2. Запускается цикл for с i от 0 до N1 - 1.
   а) Запускается цикл for с j от 0 до M1 - 1 и в нем matrix_res[i][j] присваивается matrix[j][i].
3. Каждый элемент matrix_res берется по модулю mat_mod.
4. Выводится матрица matrix_res.

Временная сложность алгоритма операции транспонирования матрицы над конечным полем = $O(n * m)$, где n это N1, m это M1

2.14 Алгоритм нахождения определителя матрицы над конечным полем

Вход: Размерность матрицы $N1$, матрица matrix размерности $N1 \times N1$ и число, по модулю которого будет выполняться операция mat_mod.

Выход: Определитель матрицы matrix над конечным полем.

1. Если N1 == 2, определитель равен matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0].
2. Если N1 == 1, определитель равен matrix[0][0].
3. Если N1 >= 3
   а) Создается матрица matrix_dop размерности $N1 - 1 \times N1 - 1$
   б) Создается переменная det = 0, которая будет являться определителем матрицы matrix;
   в) Создаются переменные a и b.
   г) Запускается цикл for с j от 0 до N1 - 1
      i. a = 0;
      ii. Запускается цикл for с k от 1 до N1 - 1
         А. b = 0;
         Б. Запускается цикл for с s от 0 до N1 - 1 и если в нем s != j, то matrix_dop[a][b] присваивается matrix[k][s] и b увеличи-

вается на 1

B. a++;

iii. К det прибавляется произведение -1^{j+2} на matrix[0][j] на рекурсивный вызов данного алгоритма от matrix_dop, N1 - 1 и mat_mod.

д) Пока det < 0, выполняется det += mat_mod.

е) det = det % mat_mod;

ж) Возвращается det.

Временная сложность алгоритма нахождения определителя матрицы над конечным полем = $O(n^3)$.

2.15 Алгоритм операции обращения матрицы над конечным полем

Вход: Размерность матрицы $N1 \times M1$, матрица matrix размерности $N1 \times M1$ и число, по модулю которого будет выполняться операция mat_mod.

Выход: Матрица с выполненной операцией обращения над конечным полем.

1. Если N1 != M1, алгоритм не выполняется.
2. Создается матрица obr_matr.
3. Считается определитель матрицы matrix по алгоритму 2.14 и переменной determ присваивается найденный определитель.
4. Создается переменная obr_determ и запускается цикл for с i от 1 до 100000
   а) Если (i * determ) % mat_mod = 1, переменной obr_determ присваивается i и цикл завершается.
5. Если determ != 0, запускается цикл for с i от 0 до N1 - 1.
   а) Запускается цикл for с j от 0 до M1 - 1
      i. Создается матрица temp_matr размерности $N1 - 1 \times M1 - 1$.
      ii. Удаляются строка и столбец i, j из temp_matr.
      iii. obr_matr[i][j] присваивается частное произведения -1^{i+j+2} на определитель матрицы temp_matr и определителя determ.
      iv. Создается переменная dop_el, которая равна произведению -1 в степени i + j + 2 на определитель temp_matr размерности N1 - 1, взятый по модулю mat_mod.
      v. Пока переменная dop_el < 0, выполняется dop_el = dop_el + mat_mod.

vi. dop_el = dop_el % mat_mod;
vii. dop_el = dop_el * obr_determ;
viii. Каждому элементу obr_matr[i][j] присваивается значение dop_el % mat_mod;
ix. Запускается цикл for с k от 0 до M1 - 1 и в нем каждый раз matrix_res[i][j] увеличивается на matrix1[i][k] * matrix2[k][j].

6. Транспонируется матрица obr_matr.
7. Выводится матрица matrix_transp (транспонированная матрица obr_matr).

Временная сложность алгоритма операции обращения матрицы над конечным полем = $O(n^5)$

3 Код программы

```
#include <iostream>
#include <vector>
#include <math.h>

using namespace std;

int CharInt(int N, char c, vector <char> mnojestvo) {
for (int i = 0; i < N; i++)
{
if (mnojestvo[i] == c)
return i;
}
}

bool proverka_Ass(int N, char** keli, vector <char> mnojestvo) {
bool prov = true;
for (int x = 0; x < N; x++)
{
for (int y = 0; y < N; y++)
{
for (int z = 0; z < N; z++)
{
if (keli[x][CharInt(N, keli[y][z], mnojestvo)]
!= keli[CharInt(N, keli[x][y], mnojestvo)][z])
prov = false;
}
}
}
return prov;
}

bool proverka_Komm(int N, char** keli, vector <char> mnojestvo) {
bool prov = true;
```

```
for (int x = 0; x < N; x++)
{
for (int y = 0; y < N; y++)
{
if (keli[x][y] != keli[y][x])
prov = false;
}
}
return prov;
}


bool proverka_Idem(int N, char** keli, vector <char> mnojestvo) {
bool prov = true;
for (int x = 0; x < N; x++)
{
if (keli[x][x] != mnojestvo[x])
prov = false;
}
return prov;
}

bool proverka_Obr(int N, char** keli, vector <char> mnojestvo) {
bool prov = true;
for (int x = 0; x < N; x++)
{
for (int y = 0; y < N; y++)
{
if (keli[x][y] != 1 && keli[y][x] != 1)
prov = false;
}
}
return prov;
}
```

```
bool proverka_Dist(int N, char** keli1, char** keli2
  , vector <char> mnojestvo) {
bool prov = true;
for (int x = 0; x < N; x++)
{
for (int y = 0; y < N; y++)
{
for (int z = 0; z < N; z++)
{
if (keli1[x][CharInt(N, keli2[y][z], mnojestvo)] !=
keli2[CharInt(N, keli1[x][y], mnojestvo)][CharInt(N, keli1[x][z]
, mnojestvo)] ||
keli1[CharInt(N, keli2[y][z], mnojestvo)][x] !=
keli2[CharInt(N, keli1[y][x], mnojestvo)][CharInt(N, keli1[z][x]
, mnojestvo)])
prov = false;
}
}
}
return prov;
}

void proverka_1(int N, vector <char> mnojestvo, char** keli1
, char** keli2) {

cout << endl;
//проверка на ассоциативность
// пересечение
if (proverka_Ass(N, keli1, mnojestvo) == true)
cout << "Операция пересечения ассоциативна" << endl;
else
cout << "Операция пересечения не ассоциативна" << endl;
// объединение
```

```
if (proverka_Ass(N, keli2, mnojestvo) == true)
cout << "Операция объединения ассоциативна" << endl;
else
cout << "Операция объединения не ассоциативна" << endl;

cout << endl;
//проверка на коммутативность
// пересечение
if (proverka_Komm(N, keli1, mnojestvo) == true)
cout << "Операция пересечения коммутативна" << endl;
else
cout << "Операция пересечения не коммутативна" << endl;
// объединение
if (proverka_Komm(N, keli2, mnojestvo) == true)
cout << "Операция объединения коммутативна" << endl;
else
cout << "Операция объединения не коммутативна" << endl;

cout << endl;
//проверка на идемпотентность
// пересечение
if (proverka_Idem(N, keli1, mnojestvo) == true)
cout << "Операция пересечения идемпотентна" << endl;
else
cout << "Операция пересечения не идемпотентна" << endl;
// объединение
if (proverka_Idem(N, keli2, mnojestvo) == true)
cout << "Операция объединения идемпотентна" << endl;
else
cout << "Операция объединения не идемпотентна" << endl;

cout << endl;
//проверка на обратимость
// пересечение
```

```
if (proverka_Obr(N, keli1, mnojestvo) == true)
cout << "Операция пересечения обратима" << endl;
else
cout << "Операция пересечения не обратима" << endl;
// объединение
if (proverka_Obr(N, keli2, mnojestvo) == true)
cout << "Операция объединения обратима" << endl;
else
cout << "Операция объединения не обратима" << endl;

cout << endl;
//проверка на дистрибутивность
if (proverka_Dist(N, keli1, keli2, mnojestvo) == true)
cout << "Операция дистрибутивна" << endl;
else
cout << "Операция не дистрибутивна" << endl;

cout << endl;
}

void bin_otn_two(int N, bool** matrix1, bool** matrix2, int bo_vvod) {

cout << endl;
if (bo_vvod == 1) {     //операция объединения
bool** matrix_res;
matrix_res = new bool* [N];
for (int i = 0; i < N; i++) {
matrix_res[i] = new bool[N];
for (int j = 0; j < N; j++) {
if (matrix1[i][j] == 1 || matrix2[i][j] == 1)
matrix_res[i][j] = 1;
else
matrix_res[i][j] = 0;
}
```

```
}
cout << "Итоговая матрица: " << endl << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
cout << matrix_res[i][j] << " ";
}
cout << endl;
}
}
else if (bo_vvod == 2) { //операция пересечения
bool** matrix_res;
matrix_res = new bool* [N];
for (int i = 0; i < N; i++) {
matrix_res[i] = new bool[N];
for (int j = 0; j < N; j++) {
if (matrix1[i][j] == 0 || matrix2[i][j] == 0)
matrix_res[i][j] = 0;
else
matrix_res[i][j] = 1;
}
}
cout << "Итоговая матрица: " << endl << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
cout << matrix_res[i][j] << " ";
}
cout << endl;
}
}
else if (bo_vvod == 4) { //операция произведения
bool** matrix_res;
matrix_res = new bool* [N];
for (int i = 0; i < N; i++) {
matrix_res[i] = new bool[N];
```

```
for (int j = 0; j < N; j++) {
matrix_res[i][j] = 0;
for (int k = 0; k < N; k++) {
if (matrix1[i][k] == 1 && matrix2[k][j] == 1)
matrix_res[i][j] += 1;
}
}
}
cout << "Итоговая матрица: " << endl << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
cout << matrix_res[i][j] << " ";
}
cout << endl;
}
}
}

void bin_otn_one(int N, bool** matrix, int bo_vvod) {

cout << endl;
if (bo_vvod == 3) {     //операция дополнения
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
if (matrix[i][j] == 1)
matrix[i][j] = 0;
else
matrix[i][j] = 1;
}
}
cout << "Итоговая матрица: " << endl << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
cout << matrix[i][j] << " ";
```

```
}
cout << endl;
}
}
else if (bo_vvod == 5) { //операция нахождения обратного
bool** matrix_res;
matrix_res = new bool* [N];
for (int i = 0; i < N; i++) {
matrix_res[i] = new bool[N];
for (int j = 0; j < N; j++) {
matrix_res[i][j] = matrix[j][i];
}
}
cout << "Итоговая матрица: " << endl << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
cout << matrix_res[i][j] << " ";
}
cout << endl;
}
}
}

void matr_two(int N1, int M1, int** matrix1, int N2, int M2
  , int** matrix2, int mat_vvod, int mat_mod) {

cout << endl;
if (mat_vvod == 1) {     // сложение матриц
if (N1 != N2 || M1 != M2) {
cout << "Нельзя выполнить операцию сложения" << endl;
return;
}
else {
int** matrix_res;
```

```
matrix_res = new int* [N1];
for (int i = 0; i < N1; i++) {
matrix_res[i] = new int[M1];
for (int j = 0; j < M1; j++) {
matrix_res[i][j] = matrix1[i][j] + matrix2[i][j];
}
}
cout << "Итоговая матрица: " << endl << endl;
for (int i = 0; i < N1; i++) {
for (int j = 0; j < M1; j++) {
cout << matrix_res[i][j] % mat_mod << " ";
}
cout << endl;
}
}


}
else if (mat_vvod == 2) {      // умножение матриц
if (M1 != N2) {
cout << "Нельзя выполнить операцию умножения" << endl;
return;
}
else {
int** matrix_res;
matrix_res = new int* [N1];
for (int i = 0; i < N1; i++) {
matrix_res[i] = new int[M2];
for (int j = 0; j < M2; j++) {
matrix_res[i][j] = 0;
for (int k = 0; k < M1; k++)
matrix_res[i][j] += matrix1[i][k] * matrix2[k][j];
}
}
cout << "Итоговая матрица: " << endl << endl;
```

```
for (int i = 0; i < N1; i++) {
for (int j = 0; j < M2; j++) {
cout << matrix_res[i][j] % mat_mod << " ";
}
cout << endl;
}
}
}
}

int opred(int** matrix, int N1, int mat_mod) {
if (N1 == 2) {
return (matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]);
}
else if (N1 == 1)
return matrix[0][0];
else if (N1 >= 3) {
int** matrix_dop;
matrix_dop = new int* [N1 - 1];
for (int i = 0; i < N1 - 1; i++) {
matrix_dop[i] = new int[N1 - 1];
}
int det = 0;
int a, b;

for (int j = 0; j < N1; j++) {
a = 0;
for (int k = 1; k < N1; k++) {
b = 0;
for (int s = 0; s < N1; s++) {
if (s != j) {
matrix_dop[a][b] = matrix[k][s];
b++;
}
```

```
}
a++;
}
det += pow(-1, j + 2) * matrix[0][j]
* opred(matrix_dop, N1 - 1, mat_mod);
}
while (det < 0)
det += mat_mod;
det = det % mat_mod;
return det;
}
}

void delete_str_sto(int** matr, int n, int** temp_matr, int str
, int sto)
{
int ki = 0;
for (int i = 0; i < n; i++) {
if (i != str) {
for (int j = 0, kj = 0; j < n; j++) {
if (j != sto) {
temp_matr[ki][kj] = matr[i][j];
kj++;
}
}
ki++;
}
}
}

void matr_one(int N1, int M1, int** matrix, int mat_vvod, int mat_mod) {

cout << endl;
if (mat_vvod == 3) {     // транспонирование матрицы
```

```
int** matrix_res;
matrix_res = new int* [M1];
for (int i = 0; i < M1; i++) {
matrix_res[i] = new int[N1];
for (int j = 0; j < N1; j++) {
matrix_res[i][j] = matrix[j][i];
}
}
cout << "Транспонированная матрица: " << endl << endl;
for (int i = 0; i < M1; i++) {
for (int j = 0; j < N1; j++) {
cout << matrix_res[i][j] % mat_mod << " ";
}
cout << endl;
}
}
else if (mat_vvod == 4) {     // обращение матрицы
if (N1 != M1) {
cout << "Нельзя выполнить операцию обращения
, должна быть квадратная матрица" << endl;
return;
}
else {
double** obr_matr;
obr_matr = new double* [N1];
for (int i = 0; i < N1; i++) {
obr_matr[i] = new double[N1];
}
int determ = opred(matrix, N1, mat_mod);

int obr_determ;
for (int i = 1; i < 100000; ++i)
if ((i * determ) % mat_mod == 1)
{
```

```
obr_determ = i;
break;
}

cout << "Определитель матрицы по модулю " << mat_mod << " = "
 << determ << endl << endl;
if (determ) {
for (int i = 0; i < N1; i++) {
for (int j = 0; j < N1; j++) {
int m = N1 - 1;
int** temp_matr = new int* [m];
for (int k = 0; k < m; k++)
temp_matr[k] = new int[m];
delete_str_sto(matrix, N1, temp_matr, i, j);
int dop_el;
dop_el = pow(-1.0, i + j + 2) * opred(temp_matr, m, mat_mod);
while (dop_el < 0)
dop_el = dop_el + mat_mod;
dop_el = dop_el % mat_mod;
dop_el = dop_el * obr_determ;
obr_matr[i][j] = dop_el % mat_mod;
}
}
}
else
cout << "Определитель матрицы = 0, матрица вырожденная
 и обратной матрицы не имеет" << endl;


int** matrix_transp;
matrix_transp = new int* [N1];
for (int i = 0; i < N1; i++) {
matrix_transp[i] = new int[N1];
for (int j = 0; j < N1; j++) {
```

```
matrix_transp[i][j] = obr_matr[j][i];
}
}
cout << "Обратная матрица по модулю " << mat_mod << " : " << endl;
for (int i = 0; i < N1; i++) {
for (int j = 0; j < N1; j++) {
cout << matrix_transp[i][j] << " ";
}
cout << endl;
}

//ПРОВЕРКА
cout << "Перемножим исходную матрицу и обратную, получим: " << endl;
matr_two(N1, N1, matrix, N1, N1, matrix_transp, 2, mat_mod);
}

}
}

int main()
{
setlocale(LC_ALL, "Rus");
vector <char> mnojestvo;
int sposob, i, j, N;
cout << "Введите, что хотите сделать: " << endl;
cout << "1 - проверить свойства операций " << endl;
cout << "2 - выполнить операции над бинарными отношениями" << endl;
cout << "3 - выполнить операции над матрицами (над конечным полем)"
<< endl;
cin >> sposob;

if (sposob == 1)
{
cout << "Введите размерность: " << endl;
```

```
cin >> N;
if (N == 0) {
cout << "Ошибка";
return 0;
}
cout << "Введите множество: " << endl;
char vv;
for (int i = 0; i < N; i++) {
cin >> vv;
mnojestvo.push_back(vv);
}
char** keli1;
keli1 = new char* [N];
cout << "Таблица Кэли операции пересечения: " << endl;
for (int i = 0; i < N; i++) {
keli1[i] = new char[N];
for (int j = 0; j < N; j++) {
cin >> keli1[i][j];
}
}
char** keli2;
keli2 = new char* [N];
cout << "Таблица Кэли операции объединения: " << endl;
for (int i = 0; i < N; i++) {
keli2[i] = new char[N];
for (int j = 0; j < N; j++) {
cin >> keli2[i][j];
}
}
proverka_1(N, mnojestvo, keli1, keli2);
}
else if (sposob == 2)
{
cout << "Выберите операцию для бинарных отношений: " << endl;
```

```
cout << "1 - объединение, 2 - пересечение
, 3 - дополнение, 4 - произведение
, 5 - нахождение обратного" << endl;
int bo_vvod;
cin >> bo_vvod;

cout << "Введите размерность : " << endl;
cin >> N;
if (N == 0) {
cout << "Ошибка";
return 0;
}
if (bo_vvod == 1 || bo_vvod == 2 || bo_vvod == 4) {
cout << "Введите матрицу бинарного отношения №1: " << endl;
bool** matrix1;
matrix1 = new bool* [N];
for (int i = 0; i < N; i++) {
matrix1[i] = new bool[N];
for (int j = 0; j < N; j++) {
cin >> matrix1[i][j];
}
}
cout << "Введите матрицу бинарного отношения №2: " << endl;
bool** matrix2;
matrix2 = new bool* [N];
for (int i = 0; i < N; i++) {
matrix2[i] = new bool[N];
for (int j = 0; j < N; j++) {
cin >> matrix2[i][j];
}
}

bin_otn_two(N, matrix1, matrix2, bo_vvod);
}
```

```
else if (bo_vvod == 3 || bo_vvod == 5) {
cout << "Введите матрицу бинарного отношения: " << endl;
bool** matrix;
matrix = new bool* [N];
for (int i = 0; i < N; i++) {
matrix[i] = new bool[N];
for (int j = 0; j < N; j++) {
cin >> matrix[i][j];
}
}

bin_otn_one(N, matrix, bo_vvod);
}
else
cout << "Ошибка" << endl;
}
else if (sposob == 3) {
// требуется рассмотреть операции сложения, умножения
, транспонирования и обращения матриц над конечным полем.
cout << "Выберите операцию для матриц: " << endl;
cout << "1 - сложение, 2 - умножение
, 3 - транспонирование, 4 - обращение" << endl;
int mat_vvod;
int N1, M1, N2, M2;
cin >> mat_vvod;
int mat_mod;
cout << "Введите число, по модулю которого
 будет выполняться операция: " << endl;
cin >> mat_mod;
if (mat_vvod == 1 || mat_vvod == 2) {
cout << "Введите размерность матрицы №1 (N M): " << endl;
cin >> N1 >> M1;
cout << "Введите матрицу №1: " << endl;
int** matrix1;
```

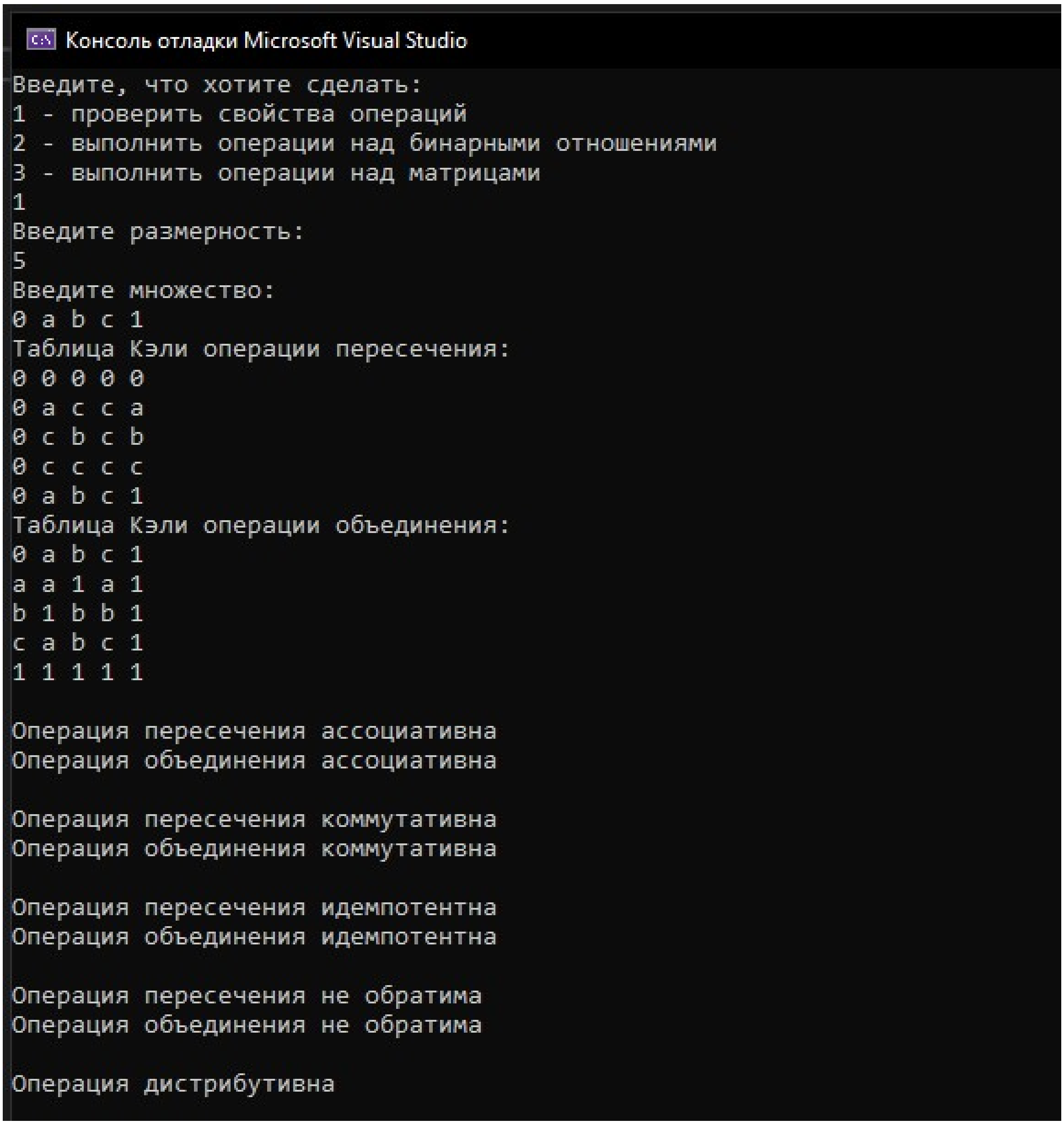
```
matrix1 = new int* [N1];
for (int i = 0; i < N1; i++) {
matrix1[i] = new int[M1];
for (int j = 0; j < M1; j++) {
cin >> matrix1[i][j];
}
}
cout << "Введите размерность матрицы №2 (N M): " << endl;
cin >> N2 >> M2;
cout << "Введите матрицу №2: " << endl;
int** matrix2;
matrix2 = new int* [N2];
for (int i = 0; i < N2; i++) {
matrix2[i] = new int[M2];
for (int j = 0; j < M2; j++) {
cin >> matrix2[i][j];
}
}

matr_two(N1, M1, matrix1, N2, M2, matrix2, mat_vvod, mat_mod);
}
else if (mat_vvod == 3 || mat_vvod == 4) {
cout << "Введите размерность матрицы (N M): " << endl;
cin >> N1 >> M1;
cout << "Введите матрицу: " << endl;
int** matrix;
matrix = new int* [N1];
for (int i = 0; i < N1; i++) {
matrix[i] = new int[M1];
for (int j = 0; j < M1; j++) {
cin >> matrix[i][j];
}
}
```

```
matr_one(N1, M1, matrix, mat_vvod, mat_mod);
}
else
cout << "Ошибка" << endl;
}
else
cout << "Ошибка" << endl;

cout << endl;
}

}
```

4 Результаты тестирования программ

Тестирование №1:

Проверка свойств операций пересечения и объединения.



```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами
1
Введите размерность:
5
Введите множество:
0 a b c 1
Таблица Кэли операции пересечения:
0 0 0 0 0
0 a c c a
0 c b c b
0 c c c c
0 a b c 1
Таблица Кэли операции объединения:
0 a b c 1
a a 1 a 1
b 1 b b 1
c a b c 1
1 1 1 1 1

Операция пересечения ассоциативна
Операция объединения ассоциативна

Операция пересечения коммутативна
Операция объединения коммутативна

Операция пересечения идемпотентна
Операция объединения идемпотентна

Операция пересечения не обратима
Операция объединения не обратима

Операция дистрибутивна
```

Рисунок 1 – Тестировние №1

Тестирование №2:

Операция объединения над бинарными отношениями.

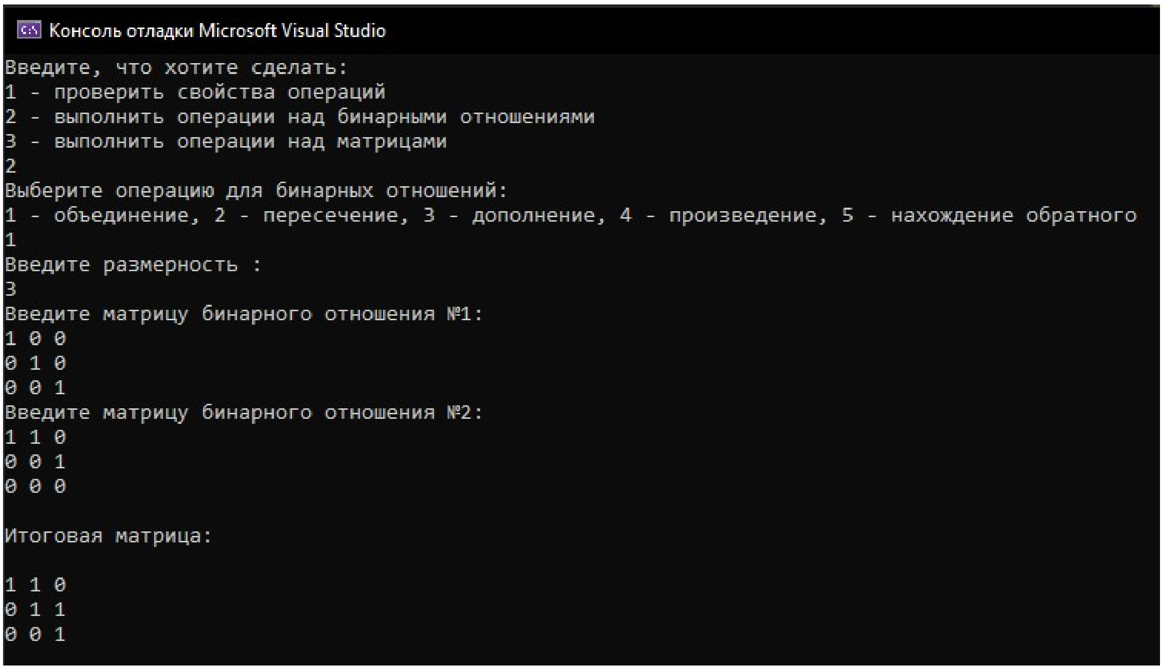


Рисунок 2 – Тестировние №2

Тестирование №3:

Операция пересечения над бинарными отношениями.

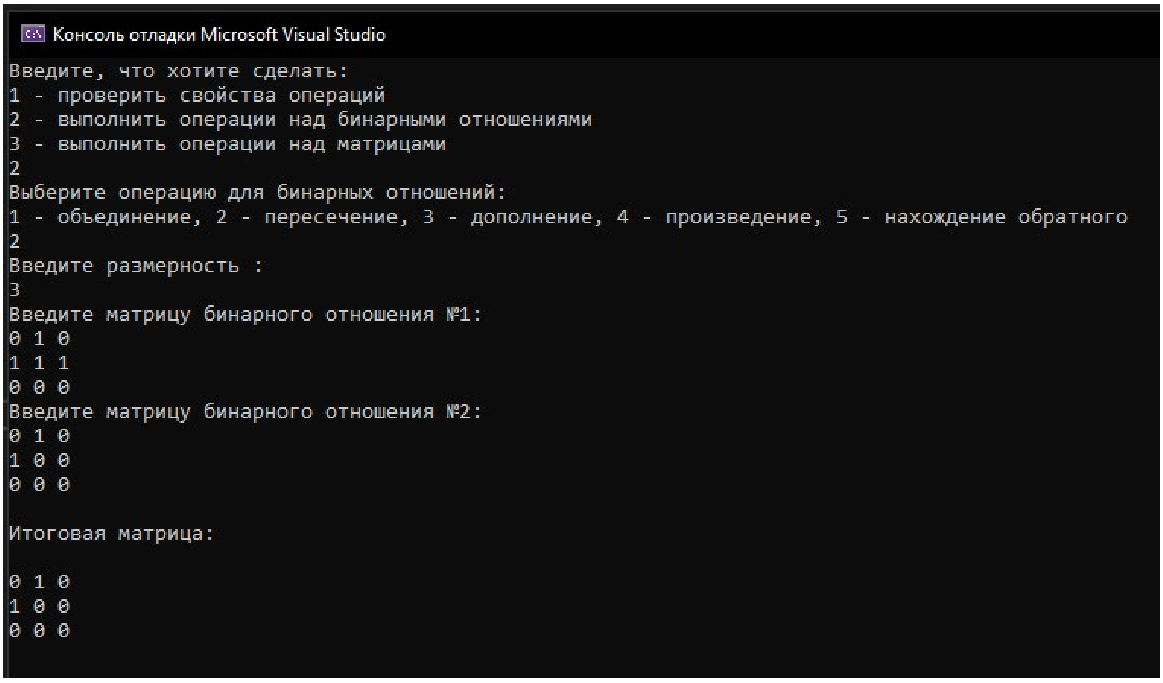
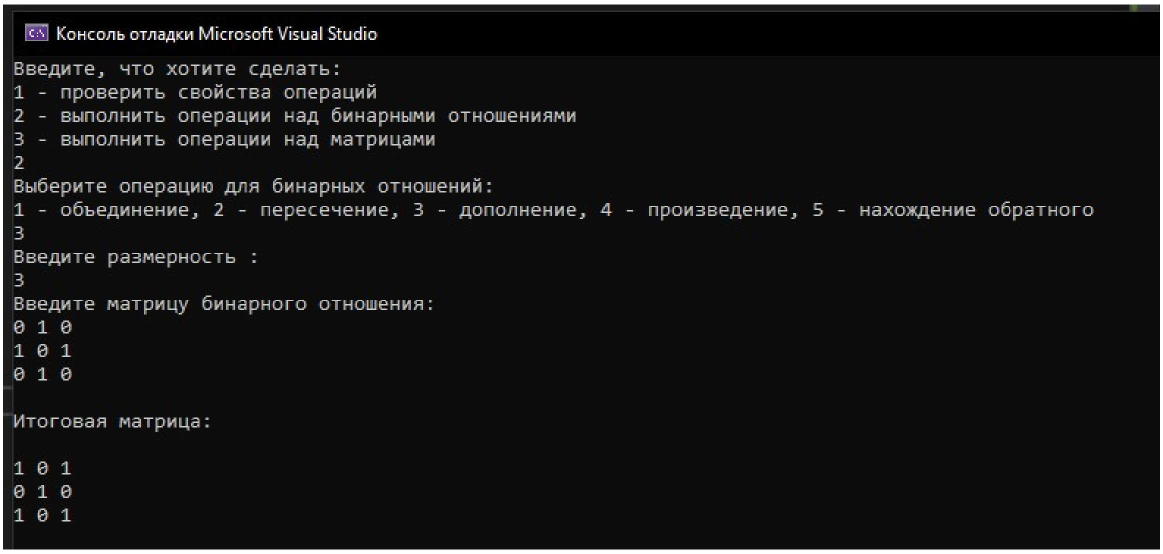


Рисунок 3 – Тестировние №3

Тестирование №4:

Операция дополнения над бинарным отношением.

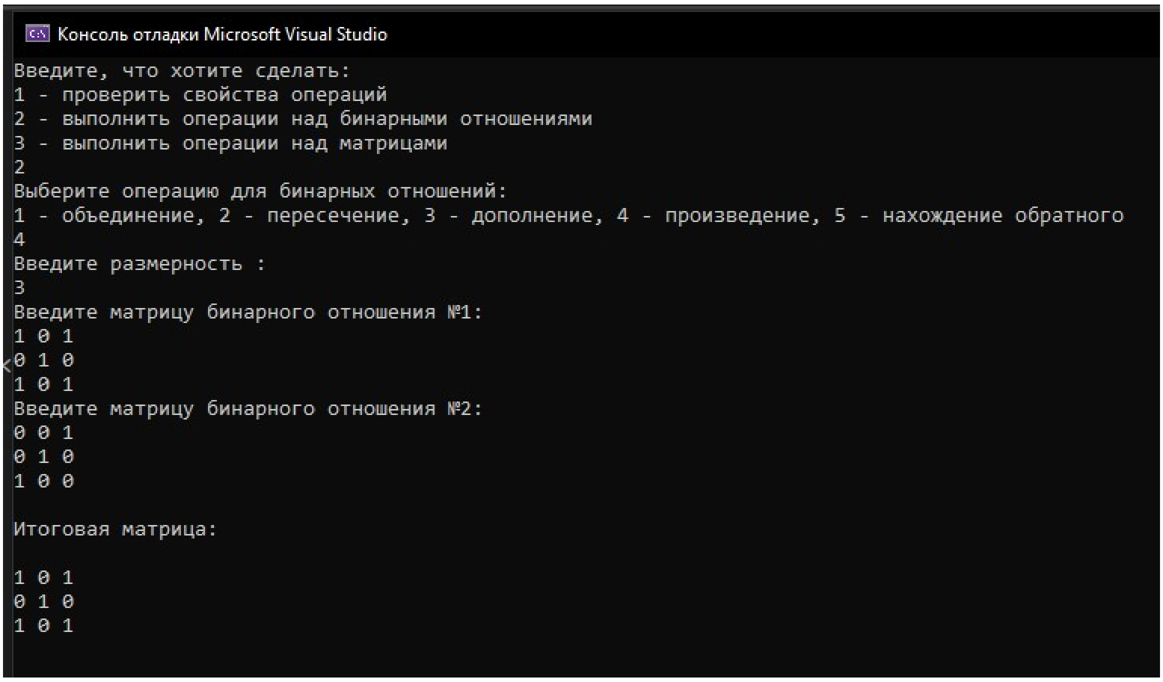


```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами
2
Выберите операцию для бинарных отношений:
1 - объединение, 2 - пересечение, 3 - дополнение, 4 - произведение, 5 - нахождение обратного
3
Введите размерность :
3
Введите матрицу бинарного отношения:
0 1 0
1 0 1
0 1 0

Итоговая матрица:

1 0 1
0 1 0
1 0 1
```

Рисунок 4 – Тестировние №4

Тестирование №5:

Операция произведения над бинарными отношениями.

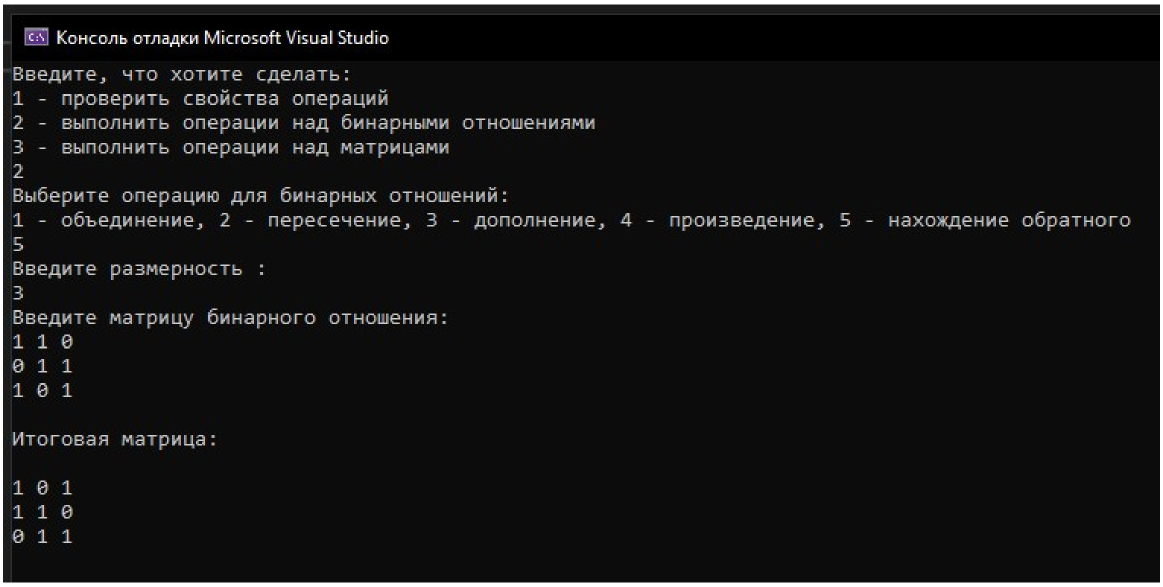


```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами
2
Выберите операцию для бинарных отношений:
1 - объединение, 2 - пересечение, 3 - дополнение, 4 - произведение, 5 - нахождение обратного
4
Введите размерность :
3
Введите матрицу бинарного отношения №1:
1 0 1
0 1 0
1 0 1
Введите матрицу бинарного отношения №2:
0 0 1
0 1 0
1 0 0

Итоговая матрица:

1 0 1
0 1 0
1 0 1
```

Рисунок 5 – Тестировние №5

Тестирование №6:

Операция нахождения обратного над бинарным отношением.

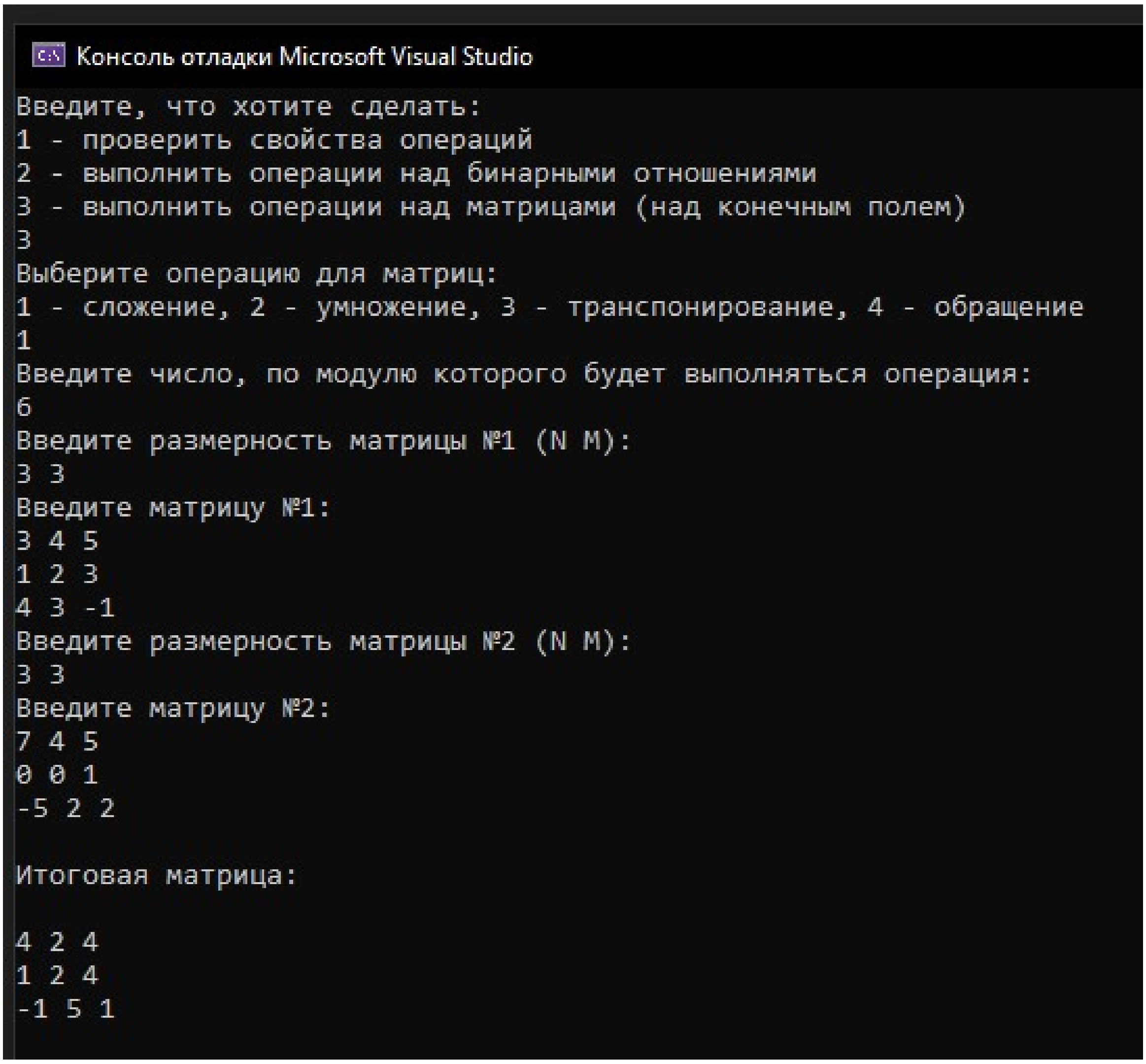


```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами
2
Выберите операцию для бинарных отношений:
1 - объединение, 2 - пересечение, 3 - дополнение, 4 - произведение, 5 - нахождение обратного
5
Введите размерность :
3
Введите матрицу бинарного отношения:
1 1 0
0 1 1
1 0 1

Итоговая матрица:

1 0 1
1 1 0
0 1 1
```

Рисунок 6 – Тестировние №6

Тестирование №7:

Операция сложения матриц над конечным полем.




```
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами (над конечным полем)
3
Выберите операцию для матриц:
1 - сложение, 2 - умножение, 3 - транспонирование, 4 - обращение
1
Введите число, по модулю которого будет выполняться операция:
6
Введите размерность матрицы №1 (N M):
3 3
Введите матрицу №1:
3 4 5
1 2 3
4 3 -1
Введите размерность матрицы №2 (N M):
3 3
Введите матрицу №2:
7 4 5
0 0 1
-5 2 2

Итоговая матрица:

4 2 4
1 2 4
-1 5 1
```

Рисунок 7 – Тестировние №7

Тестирование №8:

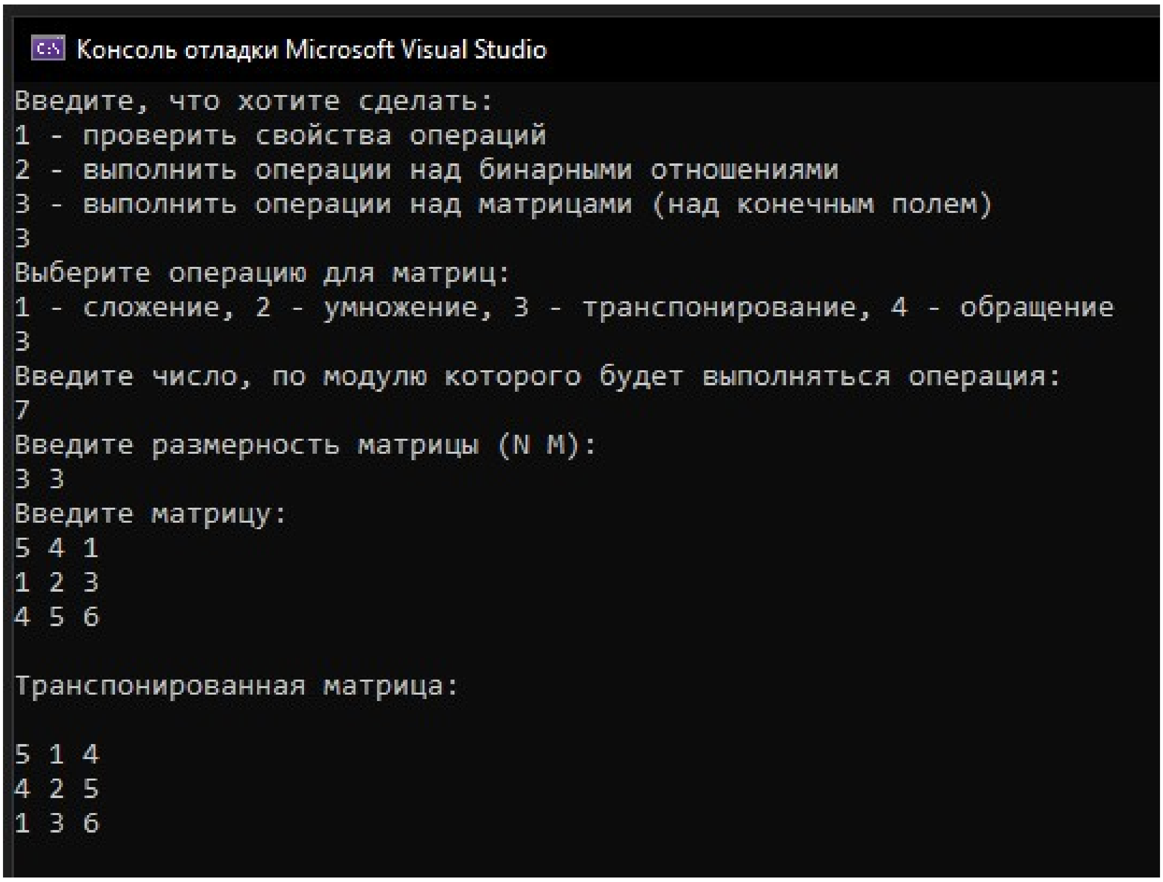
Операция умножения матриц над конечным полем.

```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами (над конечным полем)
3
Выберите операцию для матриц:
1 - сложение, 2 - умножение, 3 - транспонирование, 4 - обращение
2
Введите число, по модулю которого будет выполняться операция:
8
Введите размерность матрицы №1 (N M):
3 3
Введите матрицу №1:
4 6 1
4 5 2
2 2 1
Введите размерность матрицы №2 (N M):
3 3
Введите матрицу №2:
3 2 1
2 4 1
-7 3 5

Итоговая матрица:

1 3 7
0 2 3
3 7 1
```

Рисунок 8 – Тестировние №8

Тестирование №9:

Операция транспонирования матрицы над конечным полем.



```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами (над конечным полем)
3
Выберите операцию для матриц:
1 - сложение, 2 - умножение, 3 - транспонирование, 4 - обращение
3
Введите число, по модулю которого будет выполняться операция:
7
Введите размерность матрицы (N M):
3 3
Введите матрицу:
5 4 1
1 2 3
4 5 6

Транспонированная матрица:

5 1 4
4 2 5
1 3 6
```

Рисунок 9 – Тестировние №9

Тестирование №10:

Операция обращения матрицы над конечным полем.

```
Консоль отладки Microsoft Visual Studio
Введите, что хотите сделать:
1 - проверить свойства операций
2 - выполнить операции над бинарными отношениями
3 - выполнить операции над матрицами (над конечным полем)
3
Выберите операцию для матриц:
1 - сложение, 2 - умножение, 3 - транспонирование, 4 - обращение
4
Введите число, по модулю которого будет выполняться операция:
37
Введите размерность матрицы (N M):
3 3
Введите матрицу:
11 15 12
15 2 15
17 15 19

Определитель матрицы по модулю 37 = 7

Обратная матрица по модулю 37 :
5 22 34
1 6 18
22 34 8
Перемножим исходную матрицу и обратную, получим:

Итоговая матрица:

1 0 0
0 1 0
0 0 1
```

Рисунок 10 – Тестировние №10

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были рассмотрены и изучены следующие темы: понятие алгебраической операции и классификация свойств операций, основные операции над бинарными отношениями и основные операции над матрицами. В третьей части работы были реализованы алгоритмы проверки свойств операций: ассоциативность, коммутативность, идемпотентность, обратимость, дистрибутивность, алгоритмы выполнения операции над бинарными отношениями и алгоритмы выполнения операций над матрицами.